SwainOS

Business Command Center

Project Specification & Architecture Document

Version 1.0 | February 2026

Prepared for: Ian Swain II

CONFIDENTIAL

Table of Contents

1. Executive Summary

SwainOS is a comprehensive, AI-first business intelligence platform designed specifically for Swain Destinations. The system provides real-time visibility into business health, cash position, debt obligations, and operational metrics while delivering intelligent recommendations—particularly around foreign exchange timing, which represents a significant profit center for the business.

The platform consolidates data from multiple sources (Salesforce, QuickBooks, FX providers, banking) into a unified command center, enabling data-driven decision-making and proactive business management.

1.1 Key Objectives

- Centralize all business data into a single source of truth
- Provide real-time cash flow forecasting and debt service tracking
- Deliver AI-powered FX buy signals to optimize currency purchases
- Enable natural language queries across all business data
- Track performance against budget, loan covenants, and industry benchmarks

1.2 Technology Decisions

Decision	Choice	Rationale
Backend Framework	Python / FastAPI	Strong data science ecosystem, async support, OpenAI integration
Frontend Framework	React / Next.js	Component-based, excellent developer experience, SSR capable
Database	Supabase (PostgreSQL)	Managed PostgreSQL with built-in auth, RLS, real-time subscriptions, edge functions
AI Provider	OpenAI GPT-4	Best reasoning capabilities, function calling, production-ready
Salesforce Edition	Unlimited	Full API access, custom object support
Accounting	QuickBooks Online	Modern API, real-time sync capabilities
FX Data	Free tier initially	Alpha Vantage or exchangerate.host, upgrade path available
Hosting	Supabase Cloud + Local Dev	Supabase cloud for production, local Supabase CLI for development

2. Project Overview

2.1 Business Context

Swain Destinations is a luxury travel operator specializing in Australia, New Zealand, and Africa. The business operates on a commission model, earning revenue from bookings placed through travel advisor partners. Key financial characteristics include:

- Commission Income: ~$5.9M annually (2025)
- Net Income: ~$1.66M annually
- High cash balances due to customer deposits (liability)
- Significant foreign currency exposure (AUD, NZD, ZAR)
- SBA loan debt service: ~$998K annually

2.2 Problem Statement

Currently, business data is siloed across multiple systems with no unified view. Key challenges include:

- No real-time visibility into cash position vs. customer deposit liabilities
- Manual FX purchasing decisions without systematic analysis
- Debt service tracking via spreadsheets
- No automated alerting for business anomalies
- Limited ability to forecast cash flow accurately

2.3 Solution Overview

SwainOS addresses these challenges by providing:

- Unified data layer with scheduled ETL from all sources
- Real-time dashboards with AI-generated insights
- Algorithmic FX buy signals based on exposure, rates, and macro factors
- Automated debt service tracking with DSCR monitoring
- Natural language interface for ad-hoc queries

3. Technical Architecture

3.1 System Architecture

The system follows a three-tier architecture designed for local deployment with cloud migration capability:

Presentation Layer

- React/Next.js single-page application
- Tailwind CSS for styling
- Recharts for data visualization
- WebSocket for real-time updates

Application Layer

- FastAPI Python backend
- Async request handling
- OpenAI integration for AI features
- Celery for background job processing
- Redis for caching and job queue

Data Layer (Supabase)

- Supabase PostgreSQL database (managed)
- Row Level Security (RLS) for data access control
- Supabase Realtime for live dashboard updates
- Supabase Edge Functions for serverless API endpoints
- Supabase Auth for user authentication

3.2 Integration Architecture

All external integrations follow a common pattern:

- Dedicated service class per integration
- Scheduled sync jobs (configurable frequency)
- Internal data model (not dependent on source schema)
- Mapping layer to transform source data to internal model
- Audit logging for all sync operations

Salesforce Authentication: JWT Bearer flow with dedicated integration user. Requires: Connected App in Salesforce with digital certificate, private key stored securely in environment variables, dedicated "SwainOS Integration" user with appropriate permissions.

Sync Failure Handling: (1) Log failure to sync_logs with error details. (2) Retry up to 3 times with exponential backoff. (3) After 3 failures, send email alert to configured addresses. (4) Continue with next sync cycle (non-blocking). Manual "Sync Now" button available in Settings UI.

Data Direction: One-way sync from source systems to SwainOS (read-only). Imported data is not modified in SwainOS. All historical data synced on initial load.

4. Data Sources & Integrations

Source	Data	Sync Frequency	Integration Method
Salesforce	Bookings, itineraries, customers, deposits, pipeline	Hourly	REST API (Bulk for initial)
QuickBooks Online	Expenses, P&L, bank transactions, AP/AR	Daily	OAuth REST API
FX Provider	AUD, NZD, ZAR rates (live + historical)	15 minutes	REST API
News/Macro	Economic indicators, central bank signals	Daily	News API + OpenAI
Citizens Bank	Account balances, transactions	Manual (future API)	Manual entry / CSV import
Manual Inputs	Budget, debt schedule, targets	As needed	Admin UI

4.1 Internal Data Model

SwainOS maintains its own data model independent of source systems. This provides:

- Freedom to change source systems without application impact
- Consistent data structure regardless of source
- Historical data retention beyond source system limits
- Custom calculated fields and aggregations

4.2 Mapping Layer

Each integration includes a mapping configuration that transforms source data to the internal model. Mappings are version-controlled YAML files that can be updated without code changes.

Salesforce/Kaptio → itineraries:

```
Id → salesforce_id | KaptioTravel__Itinerary_No__c → itinerary_number |
KaptioTravel__Status__c → itinerary_status

KaptioTravel__Start_Date__c → travel_start_date | KaptioTravel__End_Date__c →
travel_end_date

Itinerary_Countries__c → primary_country | KaptioTravel__Destination__c → primary_region

KaptioTravel__Group_Size__c → pax_count | Lead_Adults__c → adult_count | Lead_Children__c
→ child_count

KaptioTravel__Itinerary_Amount__c → gross_amount | KaptioTravel__Itinerary_Cost__c →
net_amount

KaptioTravel__CommissionTotal__c → commission_amount | KaptioTravel__TotalDepositPaid__c
→ deposit_received

KaptioTravel__Outstanding__c → balance_due | KaptioTravel__Account__c → agency_id (FK
lookup)
```

Salesforce/Kaptio ItineraryBooking → bookings:

```
Id → salesforce_id | KaptioTravel__Itinerary__c → itinerary_id |
KaptioTravel__Supplier__c → supplier_id
```

```
Name → booking_number | KaptioTravel__Name__c → service_name |
KaptioTravel__ServiceLevel__c → booking_type
KaptioTravel__DateFrom__c → service_start_date | KaptioTravel__DateTo__c →
service_end_date
KaptioTravel__TotalSalesPrice__c → gross_amount | KaptioTravel__TotalNetPrice__c →
net_amount | KaptioTravel__SupplierCurrencyIsoCode__c → currency_code
```

Salesforce/Kaptio ItineraryItem → itinerary_items:

```
Id → salesforce_id | KaptioTravel__Itinerary__c → itinerary_id |
KaptioTravel__Supplier__c → supplier_id
KaptioTravel__RecordTypeName__c → item_type | KaptioTravel__Name__c → item_name
KaptioTravel__DateFrom__c → service_start_date | KaptioTravel__DateTo__c →
service_end_date
Item_Country__c → location_country | Item_Location__c → location_city
KaptioTravel__Total_Cost__c → total_cost | KaptioTravel__SupplierCurrencyIsoCode__c →
currency_code
```

Salesforce/Kaptio Payment → customer_payments:

```
Id → salesforce_id | KaptioTravel__Itinerary__c → itinerary_id | Name → payment_number
KaptioTravel__Payment_Type__c → payment_type | KaptioTravel__Method__c → payment_method
KaptioTravel__Amount__c → amount | Status__c → payment_status | Transaction_Id__c →
processor_reference
```

Salesforce/Kaptio SupplierInvoiceLine → supplier_invoice_lines:

```
Id → salesforce_id | KaptioTravel__SupplierInvoiceBooking__c → supplier_invoice_id
KaptioTravel__Itinerary__c → itinerary_id | KaptioTravel__Supplier__c → supplier_id
KaptioTravel__DueDate__c → service_date | KaptioTravel__CostDue__c → line_amount
```

Salesforce Account (Agency) → agencies:

```
Id → salesforce_id | Name → agency_name | Agency_Code__c → agency_code
Email__c → contact_email | IsActive__c → is_active
```

Salesforce Contact (Advisor) → advisors:

```
Id → salesforce_id | AccountId → agency_id (FK lookup) | Name → advisor_name | Email →
email
```

Salesforce Account (Customer) → customers:

```
Id → salesforce_id | Name → full_name | PersonEmail → email | Phone → phone |
BillingCountry → country
```

Salesforce/Kaptio Supplier → suppliers:

```
Id → salesforce_id | Name → supplier_name | Supplier_Code__c → supplier_code |
RecordType.Name → supplier_type
KaptioTravel__Currency__c → default_currency | Payment_Terms__c → payment_terms_days |
BillingCountry → address_country
```

Salesforce/Kaptio SupplierInvoice → supplier_invoices:

```
Id → salesforce_id | Name → invoice_number | KaptioTravel__Supplier__c → supplier_id
KaptioTravel__InvoiceDate__c → invoice_date | KaptioTravel__DueDate__c → due_date |
KaptioTravel__TotalAmount__c → total_amount | CurrencyIsoCode → currency_code
```

4.3 Supabase Database Schema

The Supabase project “swain-os” contains all tables organized in the public schema with Row Level Security (RLS) enabled on all tables.

Core Tables

bookings — Individual booking/reservation records (many bookings per itinerary)

```
id (uuid, PK), salesforce_id (text, unique), itinerary_id (uuid, FK),
booking_number (text), booking_type (text), supplier_id (uuid, FK nullable),
service_name (text), service_start_date (date), service_end_date (date),
location_country (text), location_city (text), pax_count (integer),
gross_amount (numeric), net_amount (numeric), commission_amount (numeric),
currency_code (text), booking_status (text), confirmation_number (text),
created_at (timestamptz), updated_at (timestamptz), synced_at (timestamptz)
```

itineraries — Master itinerary records (parent of bookings) with location hierarchy for Active Travelers Map

```
id (uuid, PK), salesforce_id (text, unique), itinerary_number (text),
itinerary_name (text), itinerary_status (text — see Status Enum below),
travel_start_date (date), travel_end_date (date), primary_country (text),
primary_region (text), primary_city (text), primary_latitude (numeric),
primary_longitude (numeric), pax_count (integer), adult_count (integer),
child_count (integer), gross_amount (numeric), net_amount (numeric),
commission_amount (numeric), deposit_received (numeric), balance_due (numeric),
currency_code (text), customer_id (uuid, FK), agency_id (uuid, FK), advisor_id
(uuid, FK), created_at (timestamptz), updated_at (timestamptz), synced_at
(timestamptz)
```

Itinerary Status Enum Values:

```
Active: Assigned, Proposal Sent, Deposited/Confirming, Pre-Departure, eDocs Sent,
Traveling, Holding
Completed: Traveled
Closed (No Revenue): Lost, Rejected, Cancelled, Cancel Fees, Amendment Merged, Amendment
Rejected
Exclude from Reporting: Duplicate Itinerary, Test Itinerary, Sample Itinerary, Snapshot
Booking
```

itinerary_items — Line items (hotels, flights, tours) for detailed reconciliation and supplier matching

```
id (uuid, PK), salesforce_id (text, unique), itinerary_id (uuid, FK),
supplier_id (uuid, FK), item_type (text), item_name (text), item_description
(text), service_start_date (date), service_end_date (date), location_country
(text), location_region (text), location_city (text), location_latitude
(numeric), location_longitude (numeric), quantity (integer), unit_cost
(numeric), total_cost (numeric), currency_code (text), confirmation_number
(text), item_status (text), created_at (timestamptz), updated_at (timestamptz),
synced_at (timestamptz)
```

customers — Customer/traveler records

```
id (uuid, PK), salesforce_id (text, unique), full_name (text), email (text),
phone (text), country (text), created_at (timestamptz), updated_at
(timestamptz)
```

agencies — Travel agency partners

```
id (uuid, PK), salesforce_id (text, unique), agency_name (text), agency_code
(text), contact_email (text), is_active (boolean), created_at (timestamptz),
updated_at (timestamptz)
```

advisors — Travel advisors within agencies

```
id (uuid, PK), salesforce_id (text, unique), agency_id (uuid, FK), advisor_name
(text), email (text), is_active (boolean), created_at (timestamptz), updated_at
(timestamptz)
```

customer_payments — Customer deposits and payments synced from CRM

```
id (uuid, PK), salesforce_id (text, unique), itinerary_id (uuid, FK),
payment_number (text), payment_type (text: Deposit, Final Payment, Refund,
Adjustment), payment_method (text: Credit Card, Wire, Check, Agency Credit),
payment_date (date), amount (numeric), currency_code (text), payment_status
(text: Pending, Received, Failed, Refunded), processor_reference (text), notes
```

```
(text), received_at (timestamptz), created_at (timestamptz), updated_at
(timestamptz), synced_at (timestamptz)
```

Critical for cash flow: deposits are liabilities until travel occurs. Sum of received payments per itinerary = deposit_received on itineraries table.

Financial Tables

transactions — Financial transactions from QuickBooks

```
id (uuid, PK), quickbooks_id (text, unique), transaction_type (text),
transaction_date (date), amount (numeric), currency_code (text), category
(text), vendor_name (text), description (text), is_reconciled (boolean),
created_at (timestamptz), updated_at (timestamptz), synced_at (timestamptz)
```

debt_schedules — SBA loan and seller note payment schedules

```
id (uuid, PK), debt_name (text), debt_type (text), original_principal
(numeric), current_balance (numeric), interest_rate (numeric), payment_amount
(numeric), payment_frequency (text), next_payment_date (date), maturity_date
(date), created_at (timestamptz), updated_at (timestamptz)
```

debt_payments — Individual debt payment records

```
id (uuid, PK), debt_schedule_id (uuid, FK), payment_date (date),
principal_amount (numeric), interest_amount (numeric), extra_principal
(numeric), balance_after (numeric), is_paid (boolean), created_at
(timestamptz), updated_at (timestamptz)
```

budgets — Monthly/annual budget targets

```
id (uuid, PK), budget_year (integer), budget_month (integer), category (text),
budgeted_amount (numeric), actual_amount (numeric), variance_amount (numeric),
created_at (timestamptz), updated_at (timestamptz)
```

Supplier Payment Tables

suppliers — Supplier/vendor accounts (hotels, ground operators, airlines)

```
id (uuid, PK), salesforce_id (text, unique), supplier_name (text),
supplier_code (text), supplier_type (text), default_currency (text),
payment_terms_days (integer), contact_email (text), contact_phone (text),
address_country (text), is_active (boolean), created_at (timestamptz),
updated_at (timestamptz)
```

supplier_invoices — Invoice headers for supplier payments

```
id (uuid, PK), salesforce_id (text, unique), invoice_number (text), supplier_id
(uuid, FK), invoice_date (date), due_date (date), total_amount (numeric),
currency_code (text), invoice_status (text), payment_status (text), paid_amount
(numeric), paid_date (date), notes (text), created_at (timestamptz), updated_at
(timestamptz), synced_at (timestamptz)
```

supplier_invoice_lines — Line items linking invoices to bookings/itineraries

```
id (uuid, PK), salesforce_id (text, unique), supplier_invoice_id (uuid, FK),
booking_id (uuid, FK nullable), itinerary_id (uuid, FK nullable), description
(text), service_date (date), quantity (integer), unit_price (numeric),
line_amount (numeric), currency_code (text), created_at (timestamptz),
updated_at (timestamptz)
```

Invoice statuses: Draft, Pending, Approved, Paid, Cancelled. Payment statuses: Unpaid, Partial, Paid.

FX Tables

fx_rates — Historical FX rate data (time-series)

```
id (uuid, PK), currency_pair (text), rate_timestamp (timestamptz), bid_rate
(numeric), ask_rate (numeric), mid_rate (numeric), source (text), created_at
(timestamptz)
```

fx_holdings — Current currency balances (computed from fx_transactions, manually adjusted)

```
id (uuid, PK), currency_code (text, unique), balance_amount (numeric),
avg_purchase_rate (numeric), total_purchased (numeric), total_spent (numeric),
last_transaction_date (date), last_reconciled_at (timestamptz), notes (text),
created_at (timestamptz), updated_at (timestamptz)
```

fx_transactions — All FX movements: buy, sell, spend, adjustment (manual entry, no bank API)

```
id (uuid, PK), currency_code (text), transaction_type (text: BUY, SELL, SPEND,
ADJUSTMENT), transaction_date (date), amount (numeric — positive for BUY,
negative for SELL/SPEND), exchange_rate (numeric nullable), usd_equivalent
(numeric nullable), balance_after (numeric), supplier_invoice_id (uuid, FK →
supplier_invoices nullable), signal_id (uuid, FK → fx_signals nullable),
reference_number (text), notes (text), entered_by (uuid, FK → app_users),
created_at (timestamptz), updated_at (timestamptz)
```

Transaction types: BUY (purchase currency), SELL (convert back to USD), SPEND (pay supplier), ADJUSTMENT (reconciliation correction)

fx_signals — Generated FX buy/wait signals

```
id (uuid, PK), currency_code (text), signal_type (text), signal_strength
(text), current_rate (numeric), avg_30d_rate (numeric), exposure_amount
(numeric), recommended_amount (numeric), reasoning (text), generated_at
(timestamptz), expires_at (timestamptz), was_acted_on (boolean), created_at
(timestamptz)
```

Reference Tables

locations — Geographic reference data for Active Travelers Map (pre-populated)

```
id (uuid, PK), country_code (text), country_name (text), region_name (text),
city_name (text), latitude (numeric), longitude (numeric), timezone (text),
is_primary_destination (boolean)
```

Pre-populated with Swain Destinations markets: Australia (Sydney, Melbourne, Brisbane, Cairns, Perth), New Zealand (Auckland, Queenstown, Wellington), South Africa (Cape Town, Johannesburg, Kruger), and other African destinations.

currencies — Supported currencies for consistency across all currency_code fields

```
id (uuid, PK), currency_code (text, unique — ISO 4217: USD, AUD, NZD, ZAR),
currency_name (text), symbol (text), decimal_places (integer default 2),
is_active (boolean), created_at (timestamptz), updated_at (timestamptz)
```

Pre-populated with: USD (US Dollar), AUD (Australian Dollar), NZD (New Zealand Dollar), ZAR (South African Rand).

System Tables

sync_logs — Integration sync history and status

```
id (uuid, PK), source_system (text), sync_type (text), started_at
(timestamptz), completed_at (timestamptz), records_processed (integer),
records_created (integer), records_updated (integer), status (text),
error_message (text), created_at (timestamptz)
```

ai_interactions — AI query and response logging

```
id (uuid, PK), user_id (uuid, FK), interaction_type (text), prompt (text),
response (text), tokens_used (integer), model_name (text), latency_ms
(integer), created_at (timestamptz)
```

app_users — Application user profiles (linked to Supabase Auth)

```
id (uuid, PK, references auth.users), email (text), full_name (text), role
(text), preferences (jsonb), created_at (timestamptz), updated_at (timestamptz)
```

app_settings — Application configuration (sync intervals, notifications, feature flags)

```
id (uuid, PK), setting_key (text, unique), setting_value (jsonb), setting_type
(text: sync, notification, feature, general), description (text), updated_by
(uuid, FK nullable), created_at (timestamptz), updated_at (timestamptz)
```

Default settings: salesforce_sync_interval_hours (2), quickbooks_sync_interval_hours (24), fx_rate_sync_interval_minutes (15), alert_email_addresses (jsonb array), sync_failure_retry_count (3), sync_failure_alert_threshold (3)

Row Level Security (RLS) Policies

All tables have RLS enabled. Policies follow a consistent naming convention:

```
{table_name}_{action}_{role} (e.g., bookings_select_authenticated)
```

Standard Policies:

- **Read-only tables** (bookings, itineraries, itinerary_items, customers, agencies, advisors, customer_payments, transactions): SELECT for authenticated users only
- **Editable tables** (debt_schedules, debt_payments, budgets, fx_holdings, fx_transactions, app_settings): Full CRUD for authenticated users with “admin” role
- **Time-series tables** (fx_rates): SELECT for authenticated, INSERT for service_role only
- **User tables** (app_users): Users can read/update their own row only (auth.uid() = id)
- **System tables** (sync_logs, ai_interactions): SELECT for authenticated, INSERT for service_role

Database Indexes

Index naming convention: idx_{table}_{column(s)}

- `idx_bookings_travel_start_date` — Active travelers queries
- `idx_bookings_booking_status` — Status filtering
- `idx_itineraries_travel_dates` — Composite index on (travel_start_date, travel_end_date)
- `idx_itineraries_status_dates` — Active travelers filtering by status and dates
- `idx_itineraries_country` — Map aggregation by country
- `idx_itinerary_items_itinerary` — Fetch items for an itinerary
- `idx_itinerary_items_supplier` — Match items to supplier invoices
- `idx_itinerary_items_service_date` — Cash flow by service date
- `idx_fx_rates_currency_timestamp` — Time-series queries by currency
- `idx_transactions_date` — Cash flow date range queries
- `idx_debt_payments_schedule_date` — Payment schedule lookups
- `idx_supplier_invoices_due_date` — Cash flow and payment scheduling
- `idx_supplier_invoices_status` — Filter by payment status
- `idx_supplier_invoices_currency` — FX exposure by currency
- `idx_supplier_invoice_lines_booking` — Link payments to bookings
- `idx_bookings_itinerary` — Fetch all bookings for an itinerary
- `idx_customer_payments_itinerary` — Sum payments per itinerary
- `idx_customer_payments_date_status` — Cash flow by payment date
- `idx_fx_transactions_currency_date` — Balance history by currency

- `idx_fx_transactions_type` — Filter by transaction type (BUY, SELL, SPEND, ADJUSTMENT)

Materialized Views for AI Analysis

Pre-computed views refreshed on schedule for fast AI queries and dashboard performance. Raw transaction data is retained for detailed analysis.

mv_monthly_revenue — Monthly revenue summary by destination, agency, status. Refreshed daily.

```
year_month, destination_country, agency_id, itinerary_count, pax_count, gross_revenue,
net_revenue, commission_earned, avg_trip_value
```

mv_rolling_metrics — Rolling 3/6/12 month averages for trend analysis. Refreshed daily.

```
metric_date, revenue_3mo_avg, revenue_12mo_avg, bookings_3mo_avg, margin_3mo_avg,
yoy_revenue_change, yoy_booking_change
```

mv_fx_exposure — Current FX exposure by currency and time horizon (30/60/90 days). Refreshed hourly.

```
currency_code, confirmed_30d, confirmed_60d, confirmed_90d, estimated_30d, estimated_60d,
estimated_90d, current_holdings, net_exposure
```

mv_active_travelers — Current and upcoming travelers for map widget. Refreshed with Salesforce sync.

```
country, region, city, latitude, longitude, status_category, traveler_count,
itinerary_count, departing_7d, returning_7d
```

mv_cash_flow_forecast — 90-day cash flow projection combining deposits, payments, supplier invoices. Refreshed daily.

```
forecast_date, expected_deposits, expected_payments_due, supplier_invoices_due,
debt_service_due, net_cash_flow, running_balance
```

5. Application Modules

5.1 Command Center (Dashboard)

The primary landing page providing at-a-glance business health:

- Today's cash position (actual vs. available after deposits)
- 30/60/90 day cash forecast
- DSCR tracker (actual vs. required 1.25x)
- AI-generated daily briefing
- Active alerts and anomalies
- Key KPIs with trend indicators
- **Active Travelers Map** — interactive world map showing current traveler locations

Active Travelers Map Widget

A prominent visual widget displaying real-time traveler status based on itinerary travel dates from Salesforce. The map provides instant visibility into current operations:

Map Features:

- Interactive world map with destination markers (Australia, New Zealand, Africa regions)
- Color-coded markers by traveler status (pulled from Salesforce itinerary status values)
- Marker size indicates traveler count at each destination
- Click-to-drill-down showing traveler details per location
- Summary KPI cards: Total Active, Departing This Week, Returning This Week

Data Source (Salesforce):

- Itinerary object with travel start/end dates
- Itinerary status field (values defined in Salesforce configuration)
- Destination/location data from itinerary line items
- Traveler count (pax) per itinerary
- Syncs with hourly Salesforce data refresh

5.2 Cash Flow

Detailed cash flow analysis and forecasting:

- Weekly and monthly cash flow projections
- Customer deposits tracker (liability vs. available)
- Supplier payment calendar
- Budget vs. actual variance analysis
- Cash runway calculator
- Scenario modeling (what-if analysis)

5.3 Debt Service

Comprehensive loan and debt tracking:

- SBA loan tracker (balance, next payment, payoff projection)

- Seller note tracker
- Extra principal payment modeling
- DSCR trend analysis
- Early payoff scenarios
- Covenant compliance monitoring

5.4 Revenue & Bookings

Revenue analytics and pipeline visibility:

- Booking pipeline by travel date
- Commission forecast
- Revenue by destination, agency, advisor
- Cancellation tracking and trends
- Year-over-year comparisons
- Seasonality analysis

5.5 FX Command

Currency management and optimization (detailed in Section 6):

- Currency exposure dashboard
- Current rates vs. historical averages
- AI-powered buy signals
- News sentiment analysis
- FX P&L tracking
- Hedge recommendations

5.6 Operations

Operational metrics and efficiency tracking:

- Advisor productivity metrics
- Booking-to-travel conversion rates
- Supplier performance scorecards
- Trip margin analysis
- Lead time analytics

5.7 AI Insights

AI-powered analysis and interaction:

- Natural language query interface
- Weekly AI summary report
- Anomaly detection and explanation
- Recommendations queue with priority
- Trend analysis and forecasting

5.8 Settings & Administration

System configuration and management:

- Integration status and health
- Manual data entry interfaces
- Sync schedule configuration
- User preferences
- Audit logs

6. FX Signal Algorithm

The FX Command module represents a key competitive advantage and profit center. The algorithm analyzes multiple factors to generate actionable buy signals.

6.1 Algorithm Inputs

Input Category	Data Points	Source
Exposure	Future supplier payments by currency and date	Salesforce
Current Holdings	Currency balances on hand	Manual / Bank API
Market Data	Current rates, 30/60/90 day averages, volatility	FX API
Macro Signals	Central bank rates, inflation, employment data	Economic API
News Sentiment	Headlines analyzed for currency impact	News API + OpenAI

FX Exposure Estimation Algorithm:

Since supplier invoices may not exist until close to travel date, estimate future exposure using:

```
1. Confirmed invoices: Sum of supplier_invoices where payment_status != 'Paid'
2. Estimated from bookings: For itineraries in 'Deposited/Confirming', 'Pre-
Departure', 'eDocs Sent' status, estimate supplier cost as % of net_amount
(configurable, default 70%)
3. Historical buffer: Add 10-15% buffer based on historical variance between
estimated and actual
4. Closing probability: Weight by itinerary status (Deposited = 95%, Proposal
Sent = 40%, Assigned = 20%)
5. Group by currency and payment window (30/60/90 days from travel_start_date)
```

6.2 Signal Logic

For each currency (AUD, NZD, ZAR), the algorithm evaluates:

```
1. Calculate total exposure: Sum of payables in next 30/60/90 days
2. Assess rate position: Current rate vs. moving averages
3. Evaluate macro environment: Central bank stance (dovish/hawkish)
4. Analyze news sentiment: AI classification of recent headlines
5. Generate signal: BUY NOW / WAIT / HEDGE
6. Calculate optimal amount: Based on exposure, holdings, cash
available
```

6.3 Signal Output Example

The system generates human-readable recommendations:

BUY SIGNAL: AUD at 0.6523 is 2.8% below 30-day average. You have $243K AUD payable for April travel. Recommend purchasing $150K AUD now, hold $93K for potential further dip. RBA meeting next week — no rate change expected.

6.4 P&L Tracking

The system tracks FX decisions to measure performance:

- Purchase rate vs. spot rate at payment date
- Cumulative savings/cost from timing decisions
- Hit rate on buy signals
- Comparison to simple averaging strategy

7. AI Architecture

7.1 OpenAI Integration

The system uses OpenAI GPT-4 for multiple AI capabilities:

Daily Briefing Generation

- Summarizes key metrics and changes
- Highlights anomalies requiring attention
- Provides prioritized action items

Natural Language Queries

- Converts user questions to database queries
- Returns formatted, contextual responses
- Maintains conversation context

News Sentiment Analysis

- Classifies headlines by currency impact
- Extracts relevant economic signals
- Provides confidence scores

Anomaly Explanation

- Analyzes detected anomalies
- Provides possible causes
- Suggests investigation steps

7.2 AI Safety & Cost Management

- All AI calls are logged with tokens used
- Rate limiting to prevent runaway costs
- Caching of repeated queries
- Fallback to rule-based logic if API unavailable

8. Project Structure

8.1 Repository Structure

```
swain-os/
├── backend/
│   ├── api/                     # FastAPI route handlers
│   │   ├── __init__.py
│   │   ├── dashboard_routes.py
│   │   ├── cash_flow_routes.py
│   │   ├── debt_service_routes.py
│   │   ├── revenue_routes.py
│   │   ├── fx_routes.py
│   │   └── ai_routes.py
│   ├── services/                # External integrations
│   │   ├── __init__.py
│   │   ├── salesforce_service.py
│   │   ├── quickbooks_service.py
│   │   ├── fx_rate_service.py
│   │   ├── news_service.py
│   │   └── openai_service.py
│   ├── analytics/               # Business logic
│   │   ├── __init__.py
│   │   ├── cash_flow_calculator.py
│   │   ├── debt_service_tracker.py
│   │   ├── fx_signal_generator.py
│   │   ├── revenue_analyzer.py
│   │   └── anomaly_detector.py
│   ├── models/                  # Database models
│   │   ├── __init__.py
│   │   ├── booking.py
│   │   ├── transaction.py
│   │   ├── fx_rate.py
│   │   └── debt_schedule.py
│   ├── jobs/                    # Scheduled tasks
│   │   ├── __init__.py
│   │   ├── salesforce_sync_job.py
│   │   ├── quickbooks_sync_job.py
│   │   ├── fx_rate_sync_job.py
│   │   └── daily_briefing_job.py
│   ├── mappings/                # Data transformation configs
│   │   ├── salesforce_booking.yaml
│   │   ├── salesforce_itinerary.yaml
│   │   └── quickbooks_transaction.yaml
│   ├── tests/                   # Test files
│   ├── config.py
│   ├── database.py
│   └── main.py
├── frontend/
│   ├── src/
│   │   ├── pages/
│   │   ├── components/
│   │   ├── hooks/
│   │   ├── services/
│   │   └── utils/
│   ├── public/
│   └── package.json
├── data/                        # Static data files
│   ├── budget_2026.csv
│   └── debt_schedule.csv
├── docs/                        # Documentation
├── scripts/                     # Utility scripts
├── docker-compose.yml
├── .env.example
└── README.md
```

9. Naming Conventions

Consistent naming is critical for maintainability. All team members must follow these conventions strictly.

9.1 General Principles

- Use literal, descriptive names that clearly indicate purpose
- Avoid abbreviations unless universally understood (e.g., ID, URL, API)
- Names should be self-documenting; avoid needing comments to explain
- Consistency trumps personal preference

9.2 Python (Backend)

Element	Convention	Example
Files/Modules	snake_case	cash_flow_calculator.py
Classes	PascalCase	BookingRepository
Functions	snake_case, verb prefix	calculate_cash_flow()
Variables	snake_case	total_revenue
Constants	SCREAMING_SNAKE_CASE	MAX_RETRY_ATTEMPTS
Private	Leading underscore	_internal_helper()
Type hints	Always use	def get_booking(id: int) -> Booking:

9.3 TypeScript/React (Frontend)

Element	Convention	Example
Files (components)	PascalCase	CashFlowChart.tsx
Files (utilities)	camelCase	formatCurrency.ts
Components	PascalCase	DebtServiceCard
Functions	camelCase, verb prefix	fetchBookings()
Variables	camelCase	totalRevenue
Constants	SCREAMING_SNAKE_CASE	API_BASE_URL
Hooks	use prefix	useCashFlow()
Types/Interfaces	PascalCase, I prefix optional	Booking or IBooking

9.4 Database

Element	Convention	Example
Tables	snake_case, plural	bookings, fx_rates

Columns	snake_case	travel_date, commission_amount
Primary keys	id	id (SERIAL or UUID)
Foreign keys	singular_table_id	booking_id
Indexes	idx_table_column	idx_bookings_travel_date
Timestamps	created_at, updated_at	Always include both

9.5 API Endpoints

Pattern	Convention	Example
Resource	Plural nouns	/api/bookings
Single item	Resource + ID	/api/bookings/{id}
Nested	Parent/child	/api/agencies/{id}/bookings
Actions	Verb suffix (rare)	/api/reports/generate
Query params	snake_case	?travel_date_from=2026-01-01
Versioning	URL prefix	/api/v1/bookings

9.6 File Organization

- One class per file (with rare exceptions for tightly coupled small classes)
- Group by feature, not by type (e.g., /bookings contains routes, service, model)
- Tests mirror source structure (tests/analytics/test_cash_flow_calculator.py)
- Configuration separate from code (config.py, .env)

10. Coding Standards

10.1 Python Standards

- Follow PEP 8 style guide strictly
- Use Black for code formatting (line length: 100)
- Use isort for import sorting
- Use mypy for static type checking
- Docstrings required for all public functions (Google style)
- Maximum function length: 50 lines (prefer smaller)
- Maximum file length: 500 lines

Example Function

```
def calculate_debt_service_coverage_ratio(
    net_income: Decimal,
    debt_service: Decimal,
) -> Decimal:
    """
    Calculate the Debt Service Coverage Ratio (DSCR).

    Args:
        net_income: Total net income for the period.
        debt_service: Total debt service payments for the period.

    Returns:
        DSCR as a Decimal. Values >= 1.25 are considered healthy.

    Raises:
        ValueError: If debt_service is zero.
    """
    if debt_service == Decimal(0):
        raise ValueError("Debt service cannot be zero")
    return net_income / debt_service
```

10.2 TypeScript/React Standards

- Use TypeScript strict mode
- Use ESLint with Airbnb config
- Use Prettier for formatting
- Functional components only (no class components)
- Props interfaces required for all components
- Custom hooks for reusable logic
- Avoid any type; use unknown if type is truly unknown

10.3 Error Handling

- Never swallow exceptions silently
- Use custom exception classes for domain errors
- Log all errors with context
- Return meaningful error messages to frontend
- Use HTTP status codes correctly

10.4 Security

- Never commit secrets to repository
- Use environment variables for all configuration
- Validate all input data
- Sanitize data before database queries (use ORM)
- Use parameterized queries (never string concatenation)
- HTTPS only in production

10.5 Performance

- Database queries must use indexes
- Paginate all list endpoints
- Cache expensive calculations
- Async for I/O-bound operations
- Profile before optimizing

11. Testing Strategy

11.1 Testing Pyramid

Level	Coverage Target	Tools	Responsibility
Unit Tests	80%+	pytest, Jest	All business logic, utilities
Integration Tests	Key flows	pytest, Supertest	API endpoints, database
E2E Tests	Critical paths	Playwright	User journeys

11.2 Unit Testing Requirements

- All analytics functions must have unit tests
- All service methods must have unit tests
- Mock external dependencies (APIs, database)
- Test edge cases: zero, negative, null, boundary values
- Test error conditions explicitly

11.3 Integration Testing Requirements

- Test all API endpoints with real database (test DB)
- Test authentication and authorization
- Test data sync jobs with mock external APIs
- Verify database constraints and relationships

11.4 Test File Organization

```
tests/
├── unit/
│   ├── analytics/
│   │   ├── test_cash_flow_calculator.py
│   │   └── test_fx_signal_generator.py
│   └── services/
│       └── test_salesforce_service.py
├── integration/
│   ├── api/
│   │   └── test_booking_routes.py
│   └── jobs/
│       └── test_salesforce_sync_job.py
├── e2e/
│   └── test_dashboard_flow.py
├── fixtures/
│   └── sample_bookings.json
└── conftest.py
```

11.5 Test Naming Convention

Tests should clearly describe what they test:

```
def test_calculate_dscr_returns_ratio_when_inputs_valid():
def test_calculate_dscr_raises_error_when_debt_service_zero():
def test_fx_signal_returns_buy_when_rate_below_average():
```

11.6 Continuous Integration

- All tests run on every pull request
- No merge allowed with failing tests
- Coverage report generated automatically
- Linting must pass before tests run

12. Build Phases

Phase	Scope	Duration	Deliverables
1. Foundation	DB schema, UI shell, manual entry, debt tracker	2 weeks	Working local app with debt tracking
2. Cash Flow	Cash projections, budget vs actual, deposits	2 weeks	Cash flow forecasting module
3. Salesforce	SF integration, booking pipeline, revenue	2 weeks	Live booking data in system
4. FX Engine	FX rates, exposure calc, basic signals	2 weeks	FX dashboard with signals
5. AI Layer	OpenAI integration, briefings, NL queries	2 weeks	AI-powered insights
6. QuickBooks	QB integration, full P&L sync	2 weeks	Complete financial picture
7. Advanced FX	News sentiment, macro signals, refined algo	2+ weeks	Production FX system

12.1 Phase 1 Priority (First 2 Weeks)

Focus on getting a working foundation:

- Supabase project setup with core schema and RLS policies
- FastAPI backend with basic routes
- React frontend with navigation shell
- Manual data entry for debt schedule
- Basic debt service tracking dashboard
- Docker Compose for local development

12.2 Success Criteria by Phase

- Phase 1: Can track loan balance and view payment schedule
- Phase 2: Can see 90-day cash flow forecast
- Phase 3: Can see live booking pipeline from Salesforce
- Phase 4: Receives first FX buy signal
- Phase 5: Can ask natural language questions
- Phase 6: Full P&L visibility from QuickBooks
- Phase 7: FX signals include news sentiment

13. Appendix

13.1 Glossary

Term	Definition
DSCR	Debt Service Coverage Ratio - Net Income / Debt Service. Target: >= 1.25x
FX	Foreign Exchange - currency conversion
FIT	Fully Independent Travel - custom itineraries (vs. group tours)
SBA	Small Business Administration - government loan program
ETL	Extract, Transform, Load - data pipeline process
UHNW	Ultra High Net Worth - target customer segment

13.2 External API Documentation

- Salesforce REST API: https://developer.salesforce.com/docs/atlas.en-us.api_rest.meta/api_rest/
- QuickBooks Online API: https://developer.intuit.com/app/developer/qbo/docs/api/accounting/all-entities
- Alpha Vantage (FX): https://www.alphavantage.co/documentation/
- OpenAI API: https://platform.openai.com/docs/api-reference

13.3 Document Revision History

Version	Date	Author	Changes
1.0	February 2026	Ian Swain II / Claude	Initial specification

— End of Document —